Spelling Correction in Context

Guillaume Pinot, Chantal Enguehard

Spelling Correction in Context

Guillaume Pinot and **Chantal Enguehard**
Laboratoire d'Informatique de Nantes Atlantique (LINA)
Université de Nantes — 2, rue de la Houssinière — BP 92208
44322 Nantes Cedex 03 — FRANCE
http://www.sciences.univ-nantes.fr/lina/
guillaume.pinot@lina.univ-nantes.fr chantal.enguehard@univ-nantes.fr

Abstract

Spelling checkers, frequently used nowadays, do not allow to correct real-word errors. Thus, the erroneous replacement of *dessert* by *desert* is not detected. We propose in this article an algorithm based on the examination of the context of words to correct this kind of spelling errors. This algorithm uses a training on a raw corpus.

1 Introduction

Spell checkers distributed with text processing such as MS Word or OpenOffice are based on the use of a dictionary. The text is analyzed word by word: each word which does not appear in the dictionary is supposed to be erroneous so corrections are proposed to the user. Paradoxically, the performances of these checkers in error detection are degraded with the increase in the size of the dictionary because they are unable to detect real-word errors.

These real-word errors occur when one or more modifications of a word transform it into another word which is present in the dictionary.

> **example :** *This chocolate cake is a famous desert.*

The omission of an *s* in *dessert* reveals the word *desert*. This error is not detected because *desert* is present in the dictionary.

This problem was tackled during the second half of the 90's, in particular by Andrew R. Golding in (Golding 95) and (Golding & Schabes 96). He defines confusing sets (like {desert , dessert} for example) and then determines by examining the text which of these words is the best candidate. This method was used in other papers like (Jones & Martin 97) and (Mangu & Brill 97).

First, we will explain our algorithm and then, we will compare it with the method named *context word* by Andrew R. Golding (Golding 95).

2 Simultaneous Detection and Correction

Our algorithm detects and corrects the errors simultaneously.

During the examination of a word m, the algorithm compares its probability of appearing in its context with the probability that another word m' appears in the same context, m' being close to m in the sense of an arbitrary distance.

The context of a word is defined by the set of the words present in a vicinity of fixed size in number of words. Considering that it is the semantic aspect of a word which will guide the correction, we make the assumption that the order of these words is not important.

The probabilities are collected during the training part.

We now present the two distinct parts of our algorithm: the computation of the contextual probabilities and the error detection/correction process.

3 Training

The training is made on a raw corpus. This algorithm is parameterized by k: the number of words around a word that constitute its context.

3.1 Reading the Corpus

The corpus is parsed word by word. Let w_c be the current word.

3.1.1 Constitution of the Dictionary

The goal is to index all the words appearing in the corpus with their frequency.

Let D be the dictionary, composed of a set of pairs $D_i = (w_i, c_i)$, w_i being a word. Each w_i is unique. c_i is the number of occurrences of the word w_i.

The constitution of the dictionary is processed as follows:

- if D_c exists, c_c is incremented.

- else, $D_c = (w_c, 1)$ is added to D.

Thus, we obtain the number of appearances of each word in the corpus. This information will allow to calculate various probabilities thereafter.

3.1.2 Context Dictionary C

Definition The context dictionary named C gathers the co-occurrences of the words w_i and w_j, the distance between these words being lower or equal to k words. The word order is not taken into account. Each co-occurrence is supplied with its frequency $f_{i,j}$:

$$C = \{C_{i,j} \quad / \quad C_{i,j} = (\{w_i, w_j\}, f_{i,j})\}$$

Algorithm The corpus is parsed word by word. During the treatment of a word w_c, the $C_{c,j}$ with $j \in [c-k, c-1]$ are calculated. They are the k co-occurrences generated while combining w_c with the words appearing in a window of width k preceding w_c (see figure 1):

- if $C_{c,j}$ exists, $c_{c,j}$ is incremented.
- else, $C_{c,j} = (\{w_c, w_j\}, 1)$ is added to C.

We thus obtain all the 2-word sets located at a distance lower or equal to k, and their frequencies.

Complexity Complexity in space is $O(nk)$ with n the size of the corpus in number of words. In practice, it should be lower because of the redundancy of words and co-occurrences.

Let $O(f(x))$ be the complexity of the search, with x the size of the database in which the search is processed. As the size of this base can be raised by nk, complexity in time is $O(nkf(n))$.

3.2 Calculating the Probabilities

We use the data gathered during the parsing of the corpus to calculate the probabilities of the contexts kept in C.

Let B be the dictionary of the pairs of words associated with their probability. We thus have:

$$B_{i,j} = ((w_i, w_j), P(w_i|w_j))$$

$B_{i,j}$ and $B_{j,i}$ are defined for each $C_{i,j}$. The probability is calculated as follows:

$$P(w_i|w_j) = \frac{c_{i,j}}{c_j}$$

4 Detection and Correction

4.1 Similarity Between Two Words

Let $\text{edist}(w_i, w_j)$ be a function comparing two strings and returning a positive number with the condition

$$\text{edist}(w_i, w_j) = 0 \Leftrightarrow w_i = w_j$$

The largest $\text{edist}(w_i, w_j)$ is, the most distant w_i is from w_j.

The Aspell (Atkinson 05) distance function takes in account the phonetic of words so it needs a linguistic knowledge and depends on the target language. In this first version, we choose to use the minimal edit distance (Wagner & Fischer 74) which is totally independent of the target language. However, we slightly modified this function to reduce the cost of the inversion of letters.

To determine if a word is a plausible correction of the word to be corrected, we use a $\text{sim}(w_i, w_j)$ function, which takes in arguments 2 words and returns true if the 2 words are similar and false if not.

Let ϵ be the empty string, we can define sim as in figure 2.

In practice, we will take $\gamma = 8$ and $c = \text{edist}(\texttt{"a"}, \epsilon)$. These values have been determined after several experimentations.

4.2 Detection and Correction Algorithm

Let K_c be the context of a word w_c. K_c is the set of the $2k$ words located around w_c, that is to say the k words located before w_c and the k words located after w_c (see figure 3).

Let w_c be the word to correct. For each $w_j \in K_c$, we constitute the set F_j such that

$$\begin{aligned} F_j = \{B_{i,j} \in B| \\ \text{sim}(w_c, w_i) = \text{true}, \\ P(w_i|w_j) > P(w_c|w_j)\} \end{aligned}$$

We thus obtain $2k$ sets of propositions. The set of the possible propositions, F, is the union of the sets F_j (see example figure 4).

We now need a heuristic to give a score to each proposition in order to have them in a pertinent order. We here propose a first heuristic, but we are also elaborating tests to refine it.

Let G_j be the subset of F such that

$$G_j = \{B_{i,j}, B_{i,j} \in F\}$$

Let $k = 3$ and $c = 6$

We	can	have	a	lot	of	money.
w_1	w_2	w_3	w_4	w_5	w_6	w_7

w_3, w_4 and w_5 are in the context of w_6, that implies the sets $\{w_3, w_6\}$, $\{w_4, w_6\}$ and $\{w_5, w_6\}$.

Figure 1: Example of a Context During a Corpus Parsing

$$\text{sim}(w_i, w_j) = \begin{cases} \text{true if } \text{edist}(w_i, w_j) \leq \frac{\text{edist}(w_i,\epsilon)+\text{edist}(w_j,\epsilon)}{k} + c \\ \text{false else} \end{cases}$$

Figure 2: the $\text{sim}(w_i, w_j)$ definition

Let $k = 3$ and $c = 6$

We	can	see	that	this	is	a	very	good	dog.
w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}

We here have w_3, w_4, w_5, w_7, w_8 and w_9 in the context of w_6, that gives

$$K_6 = \{w_3, w_4, w_5, w_7, w_8, w_9\}$$

Figure 3: Example of a Context During the Correction

Let $k = 3$ and $w_c = \text{game}$

And

so	my	mind	game	round	to	the	business.
0.03086	0.01207		came	0.07317	0.01620	0.01571	
			same			0.00523	
			gate		0.00324	0.00305	
	0.00966		gave		0.00324	0.00174	
0.00617			name			0.00087	
			game			0.00043	

The numbers are $P(w_i|w_j)$. For example $P(\text{round}|\text{came}) = 0.07317$

$$G_1 = \{((\text{came}, \text{so}), 0.03086), ((\text{came}, \text{my}), 0.01207), ((\text{came}, \text{round}), 0.07317), ((\text{came}, \text{to}), 0.01620), ((\text{came}, \text{the}), 0.01571)\}$$

We do not detail $G_2, \ldots, G_6$ which are built in the same way.
The heuristic gives the highest score to the first proposition *came* because it appears 5 times.

For this example, we trained our algorithm on the novel *The War of the Worlds* by H. G. Wells. The sentence came from *The Time Machine* by H. G. Wells (Chapter 3) with the error added.

Figure 4: Propositions

One can then define the following H_j heuristic:

$$H_j = |G_j| + \prod_{B_{i,j} \in G_j} P(w_i|w_j)$$

This heuristic favors the propositions appearing in several sets of proposals F_j, then the strongest probabilities (see example in figure 4).

5 Comparison with Other Works

This algorithm is close to the *Context Words Method* by Andrew R. Golding (Golding 95).

5.1 The Context Words Method

In (Golding 95) like in other articles based on it ((Jones & Martin 97), (Golding & Schabes 96) and (Mangu & Brill 97)), confusion sets are used to correct real-word errors. A confusion set is a set of words which can be confused among each other, because of their close spellings ($\{\text{dessert}, \text{desert}\}$) or because they are often confused ($\{\text{between}, \text{among}\}$).

During the training, a set of $(w_c, w_i, P(w_c|w_i))$ is created (with w_c a word belonging to at least one confusion set, w_i any word). To correct a word, the probabilities of all the words of the corresponding confusion sets are computed, the highest probability being proposed in correction.

5.2 Comparison of the Two Methods

These two methods are based on the same idea: using the words present around the word to correct. They differ especially in the way of establishing sets and also in the nature of them:

- Golding supposes that its confusion sets are preestablished.

  We automatically determine words which can be confused during the correction process. This selection is based on the corpus itself and on our similarity function.

- The Golding's confusion sets are disjoint (their intersections are empty).

  This is not the case in our method: for each word w_i, we determine automatically a list of words that are similar to w_i and which occur in the same context of the words cooccurring with w_i. The list established for w_i and the list established for w_j ($w_i \neq w_j$) can encounter some words in commom.

6 Experimentations

6.1 Corpus

We would have liked to experiment our method on the same corpus as Golding in order to compare fruitfully our results with those he has obtained. Unfortunately, the Brown corpus used by Golding is not free, so we could not perform our algorithm on it.

So, our experiments on this spelling checker use the novel *les Misérables* by Victor Hugo. This corpus is divided into two parts: the training part (480588 words) and the part to be corrected (53405 words) in which errors have been added.

6.2 Experimental Method

We performe the correction and then generate the precision and the recall on the detection of error as well as the precision on correction (see figure 5 for the formulas).

6.3 Adding Errors

Real-word errors are previously added automatically without using any external resource.

This introduction is done in two steps: first the generation of the possible errors, then the introduction of these errors in the text.

6.3.1 Generation of the Possible Errors

Let D be a simple dictionary (a set of word). For each word $w_i \in D$, we associate a set of words included in D and close to w_i (in the sense of our similarity function). This method generates a base which can be used to generate real-word errors.

In practice, we use as dictionary the words appearing more than ten times in *les Misérables* to select errors using words whose context is known by our corrector. Errors on low frequency words (like apax) could not be detected in such experimentations because their context is completely unknown.

6.3.2 Adding Real-word Errors

Two parameters control the introduction of real-word errors: the density of inserted errors and the previously determined possible errors.

Errors are located using a XML tag which keeps the original word (this is the correction we wish to find).

Example: we introduce the word "game" instead of "came" in the text "And so my mind came round to the business.":

$$\text{Precision on detection} = \frac{\text{Number of words rightly detected as being erroneous}}{\text{Number of words detected as being erroneous}}$$

$$\text{Recall on detection} = \frac{\text{Number of words rightly detected as being erroneous}}{\text{Number of words rightly detected}}$$

$$\text{Precision on correction} = \frac{\text{Number of correctly corrected words}}{\text{Number of words rightly detected as being erroneous}}$$

Figure 5: Formulas of the precision and the recall

```
And so my mind
<error correction="came">game</error>
round to the business.
```

Each word of the corpus is affected or not by an error according to the probability of error fixed by the wished density.

6.4 Results

A summary of the results is given in the table 1.

Density	Precision	Recall	P. on correction
10%	0.1081926	0.9363030	0.9622054
1%	0.0206164	0.9615384	0.9500000

Table 1: Results

We note that the precision on detection is very bad. This overdetection of our algorithm is problematic.

On the other hand, we obtain a very good recall on detection and the precision of the correction is more than 95%. The correction in itself is thus very efficient.

7 Conclusion

We have presented here an algorithm that uses non-ordered contexts to detect and correct real-word errors.

The advantages of this algorithm are:

- simplicity;
- independence from any linguistic information;
- use of a raw corpus for the training;
- few parameters have to be regulated.

The disadvantages are:

- the significant size of the data generated by the training.
- the low precision on detection: the algorithm proposes corrections for a lot of correct words.

Our algorithm is intended to be used during the interactive correction of a text so its speed should be sufficient. On the other hand, the overdetection of errors constitutes a real problem.

We thus direct our research towards the definition of better heuristics of scheduling of the propositions. The size of the training corpus may also influence the quality of the results, its influence should be observed. We also plan to define ordered contexts to use the syntax in addition to semantics.

These various methods will be precisely evaluated on the same corpus to analyze their relevance.

References

(Atkinson 05) Kevin Atkinson. GNU Aspell. `http://aspell.net/`, 2005.

(Golding & Schabes 96) Andrew R. Golding and Yves Schabes. Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *Proceedings of the 34th conference on Association for Computational Linguistics*, pages 71–78. Association for Computational Linguistics, 1996.

(Golding 95) Andrew R. Golding. A bayesian hybrid method for context-sensitive spelling correction. *CoRR*, cmp-lg/9606001, 1995.

(Jones & Martin 97) Michael P. Jones and James H. Martin. Contextual spelling correction using latent semantic analysis. In *Proceedings of the fifth conference on Applied natural language processing*, pages 166–173. Morgan Kaufmann Publishers Inc., 1997.

(Mangu & Brill 97) Lidia Mangu and Eric Brill. Automatic rule acquisition for spelling correction. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 187–194. Morgan Kaufmann Publishers Inc., 1997.

(Wagner & Fischer 74) Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, 1974.